Store Language Strings in a Resource File

by Chris Barlow

Design your app for multiple languages by storing all the strings in a resource file.

Click & Retrieve Source CODE!

You might not realize it while you're knee-deep in code, but your application might be used by people who speak a different language. The average U.S. software company sells one-third of its software outside the United States. Many enterprises now operate in more than one country, so even in-house applications might be used in other countries. And if your application runs on the World Wide Web, anyone from anywhere in the world can access your Web site. Will all these users be comfortable with your menu captions, form text, and error messages?

Visual Basic makes it simple to design your application for multiple languages, allowing it to be easily used by a broad new market. The trick is to store all the strings used by your application—menu captions, control captions, toolbar tips, status-bar messages, and error messages—in a "resource" file. Visual Basic recognizes this special file when you make your project and compile it directly into the EXE.

It requires a lot more work to create localized applications than to store strings in a resource file: You must manage caption lengths and perform the actual translation. For instance, the German "Cut" menu command is "Ausschneiden"—you'll need to allow extra space on your control captions for this longer word. If you want help with the entire process, you might consider one of the products on the market designed to assist Visual Basic developers, such as VB Language Manager Pro from WhippleWare. This column, though, will give you a good start.

LOADING RESOURCE STRINGS

A resource file can store bitmaps, data, and strings. Strings are stored in "string tables,"

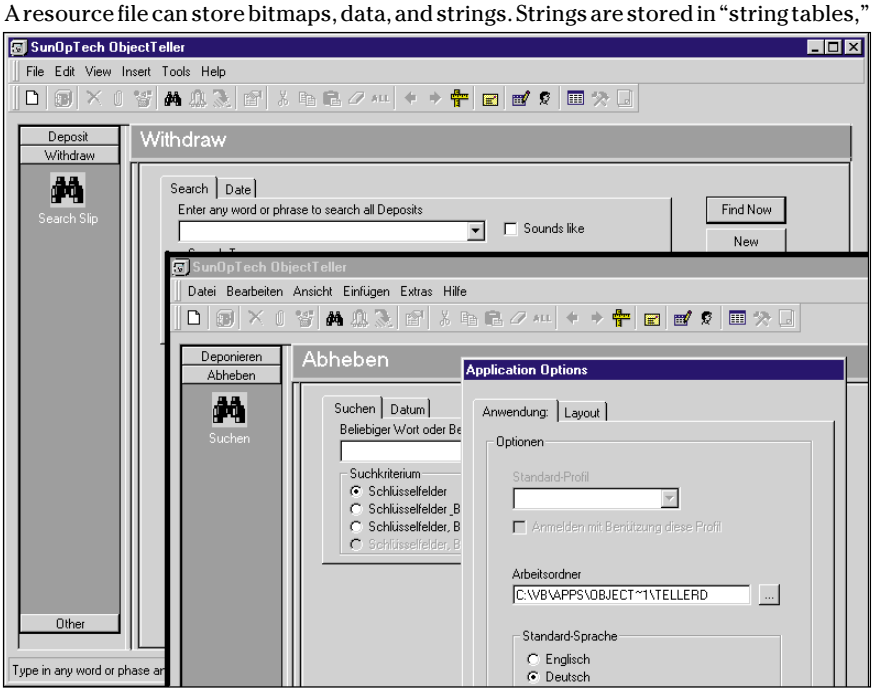

FIGURE 1 ***ObjectTeller Language Options.*** *When you change the language option, ObjectTeller refreshes all loaded forms with the new language. This screen capture shows the "before" and "after" strings on the main form.*

Chris Barlow is president of SunOpTech, a developer of document-management, decision-support, and supply-chain applications, including the ObjectBank, ObjectOrder, and ObjectJob Systems. He holds two U.S. Patents related to software for decentralized distributed asynchronous object-oriented and scheduling systems. Chris, who is a frequent speaker at VBITS, Tech•Ed, and DevDays and has been featured in two Microsoft videos, holds degrees from Harvard Business School and Dartmouth College. Reach Chris at Chris@VBExpert.com or through his Web server at http://www.VBExpert.com.

each with a unique ID. You retrieve a string from the resource file with the statement LoadResString(ID).

You can design multilanguage resource files in two ways. If you want to select the application's language strings based on the language of Windows, use a resource file with language string tables. For example, if you want someone running the German version of Windows to see your application's strings in German, you need to create a single string table with multiple language tables. However, if you want the user to be able to select your application's language independent of the Windows version, you need to create a string table with a single language table and offset the strings for each language within the string table.

For example, if the English string "File" used on your File menu has an ID of 1187, the German string "Datei" would have an ID of

VB5

```
Sub LoadResStrings(frm As Form, Optional lLanguage& = 0)
On Error Resume Next

Dim ctl As Control
Dim obj As Object
Dim obj2 As Object
Dim fnt As Object
Dim sCtlType As String
Dim nVal As Integer
Dim i As Integer
'set the form's caption
frm.Caption = GetResString(CInt(frm.Tag), lLanguage, _
 frm.Caption)

'set the font
Set fnt = frm.Font
fnt.Name = LoadResString(20)
fnt.Size = CInt(LoadResString(21))

'set the controls' captions using the caption
'property for menu items and the Tag property
'for all other controls
For Each ctl In frm.Controls
 sCtlType = TypeName(ctl)
 If sCtlType = "Label" Then
    ctl.Caption = GetResString(CInt(ctl.Tag), _
       lLanguage, ctl.Caption)
 ElseIf sCtlType = "Menu" Then
    ctl.Caption = GetResString(CInt(ctl.Caption), _
       lLanguage, ctl.Caption)
 ElseIf sCtlType = "ActiveBar" Then
    For Each obj In ctl.Bands
       For Each obj2 In obj.Tools
          If CInt(obj2.Tag) > 0 Then
          obj2.Caption = GetResString( _
             CInt(obj2.Tag), lLanguage, obj2.Caption)
          obj2.ToolTipText = _
             GetResString(CInt(obj2.Tag), _
             lLanguage, obj2.ToolTipText)
          End If
       Next
       obj.Refresh
    Next
 ElseIf sCtlType = "SSListBar" Then
    For Each obj In ctl.ListItems
       obj.Text = GetResString(CInt(obj.TagVariant), _
          lLanguage, obj.Text)
    Next
 ElseIf sCtlType = "TabStrip" Then
    For Each obj In ctl.Tabs
       obj.Caption = GetResString(CInt(obj.Tag), _
          lLanguage, obj.Caption)
       obj.ToolTipText = _
          GetResString(CInt(obj.ToolTipText), _
          lLanguage, obj.ToolTipText)
    Next
 ElseIf sCtlType = "Toolbar" Then
    For Each obj In ctl.Buttons
       obj.ToolTipText = _
          GetResString(CInt(obj.ToolTipText), _
          lLanguage, obj.ToolTipText)
    Next
 ElseIf sCtlType = "ListView" Then
    For Each obj In ctl.ColumnHeaders
       obj.Text = GetResString(CInt(obj.Tag), _
          lLanguage, obj.Text)
    Next
 ElseIf sCtlType = "GridEX" Then
    ctl.GroupByBoxInfoText = _
       GetResString(CInt(ctl.Tag), lLanguage, _
       ctl.GroupByBoxInfoText)

 ElseIf sCtlType = "SSTab" Then
    'Here use ParseString for tag CRB 18-Mar-98
    '07:26:17
    For nVal = 0 To ctl.Tabs
       ctl.TabCaption(nVal - 1) = _
          GetResString(CInt(ParseString(ctl.Tag, _
          ",", nVal)), lLanguage, _
          ctl.TabCaption(nVal - 1))
    Next
    ctl.Refresh
 ElseIf sCtlType = "CommonDialog" Then
 ElseIf sCtlType = "StatusBar" Then
'       MsgBox "status bar fontsize=" & ctl.Font.Size
 ElseIf sCtlType = "SplitFrame" Then
 ElseIf sCtlType = "ScrollPanel" Then
 ElseIf sCtlType = "PictureBox" Then
 ElseIf sCtlType = "TextBox" Then
 ElseIf sCtlType = "ComboBox" Then
 ElseIf sCtlType = "Image" Then
 ElseIf sCtlType = "ListBox" Then
 Else
    nVal = 0
    nVal = Val(ctl.Tag)
    If nVal > 0 Then ctl.Caption = _
       GetResString((nVal), lLanguage, ctl.Caption)
    nVal = 0
    nVal = Val(ctl.ToolTipText)
    If nVal > 0 Then ctl.ToolTipText = _
       GetResString((nVal), lLanguage, _
       ctl.ToolTipText)
 End If
Next
frm.Refresh
End Sub

Function GetResString(ID&, Optional lLanguage& = 0, _
 Optional OldCaption$) As String
On Error Resume Next
GetResString = LoadResString(ID + lLanguage)
If Err And lLanguage > 0 Then
 GetResString = LoadResString(ID)
 If Err Then GetResString = OldCaption
End If
On Error GoTo 0
End Function

Function GetResString(ID&, Optional lLanguage& = 0, _
 Optional OldCaption$) As String
On Error Resume Next
GetResString = LoadResString(ID + lLanguage)
If Err And lLanguage > 0 Then
 GetResString = LoadResString(ID)
 If Err Then GetResString = OldCaption
End If
On Error GoTo 0
End Function
```

LISTING 1 ***LoadResStrings Procedure.*** *Call this procedure for a form when the form is loaded or when all forms are refreshed. The lLanguage parameter is added to the resource ID to select the specified language.*

2187. You would always add 1000 to an English ID to get the German string. In one of my company's projects, I use the second method because I want the language to be independent of the Windows version. From the Options form, I select the default language and when I click on the Save button, all the strings change to that language (see Figure 1). You need to decide on a method to keep track of these unique IDs for each of your application's strings. I use constants for the text for error messages, message boxes, and status bars, and I use the Tag property for controls.

To start working with resource files, be sure to use the Professional or Enterprise edition of Visual Basic and download the Resource Editor add-in that allows you to edit string tables within the VB IDE. You can download this add-in from the Visual Basic Owner's area at http://premium.microsoft.com/download/vbasic/ResEditl.exe. This Resource Editor, contained in the ResEdit DLL, was written in Visual Basic. It reads and writes RES files directly so you can build them more easily than the old method. Previously, you needed to create a special formatted text file, called an RC file, with all your resource data, then compile the text file into a RES file with a resource compiler, typically called RC.exe.

Once you add a resource file to your Visual Basic project, your application calls a procedure when each form is loaded to load the caption or text for every control on the form from the resource file. For most controls, you can use the Tag property to store the string ID. For example, in the File menu item Tag property, you can store a value of 1187. When the form is loaded, you can load the appropriate string from the resource file with code like this:

```
mnuFile.caption = LoadResString(mnuFile.Tag + lDefLang)
```

This code sets the mnuFile caption to "File" if the default language value is zero for English, and "Datei" if the default language is 1000 for German.

You can write a procedure that steps through the Controls collection for a given form that is passed as an argument. For example, this code snippet uses the For Each statement to loop through the form's Controls collection, check the TypeName of the control, and set the Caption property if the control is a label control:

```
For Each ctl In frm.Controls
   sCtlType = TypeName(ctl)
   If sCtlType = "Label" Then
      ctl.Caption = GetResString(CInt(ctl.Tag), _
         lLanguage, ctl.Caption)
```

BAG OF TRICKS

Now you need to store the proper string ID for each control within the control. Some controls do not have a Tag property for every place you want a caption. For example, one of the commonly used Tab controls has only a single Tag property rather than one for each tab. For this control, I decided to put the IDs for all the tabs, comma-delimited, in the single Tag property.

To make it a bit more difficult, some controls, including third-party controls, have child objects that contain their own Tag properties. For these controls, you need to write some custom code. Your procedure needs to check the Tag property and load the appropriate string property for each of these child objects. For example, with the Sheridan ActiveListBar control, you might use code like this:

```
ElseIf sCtlType = "SSListBar" Then
   For Each obj In ctl.ListItems
      obj.Text = GetResString(Cint(obj.TagVariant), _
         lLanguage, obj.Text)
   Next
```

This next routine loads the strings for an individual control. I designed it around an If statement for each of my commonly used controls. You might need to modify it for your own set of controls. Notice that the Text property is filled for some controls, while the Caption property and even the ToolTipText property are used for other controls (see Listing 1).

The GetResString procedure wraps the LoadResString statement to load the proper string for each object. The LoadResString statement generates an error if the ID isn't found in the resource file. I want to load the English string if an ID for another language is missing and then return the current string if no English ID exists. This code lets me create an application in the usual manner with no resource file and build the resource file later when I have a complete set of controls and messages:

```
Function GetResString(ID&, Optional _
   lLanguage& = 0, Optional OldCaption$) As String
On Error Resume Next
GetResString = LoadResString(ID + lLanguage)
If Err And lLanguage > 0 Then
   GetResString = LoadResString(ID)
   If Err Then GetResString = OldCaption
End If
On Error GoTo 0
End Function
```

If you want to change the application's language "on the fly," you need a routine that loops through all loaded forms and refreshes the strings. This routine uses the For Each statement to loop through the Forms collection and call the LoadResStrings procedure for each form:

```
Public Sub RefreshAllFormLang(Optional lLanguage& = 0)
Dim frm As Form
For Each frm In Forms
   LoadResStrings frm, lLanguage
   frm.Refresh
Next
End Sub
```

Visual Basic's Application Wizard can create an application that uses a resource file. One of the wizard's dialogs asks if you want to keep your application strings in a resource file. Then the wizard builds all the application strings into the resource file you specify. Although the wizard uses the Tag property for the resource IDs, it doesn't fill the English strings into each control's Caption property. Instead, it puts the ID into the caption. I find this a bit confusing when working with the form in design mode, so I always enter the English caption and use it as the default. You can download the complete project from the free, Registered Level of The Development Exchange (see the Code Online box for details). ☒

Code Online

You can find all the code published in this issue of **VBPJ** *on The Development Exchange (DevX) at http://www.windx.com. For details, please see "Get Extra Code in DevX's Premier Club" in Letters to the Editor.*

Store Language Strings in a Resource File

Locator+ Codes

Listings for the entire issue, plus the application built using the Application Wizard (free Registered Level): VBPJ0798

✪ ***Listings for this article only, plus the code described above and the complete ObjectTeller RES and RC files with 300 English/German translations to help you get started quickly (subscriber Premier Level): GS0798***